

**WHAT IS CLAIMED IS:**

- 1           1.    A processor comprising:  
2                an out-of-order microinstruction pointer ( $\mu$ IP) stack in a  
3                microcode ( $\mu$ code) execution core.
- 1           2.    The processor of claim 1 in which the  $\mu$ IP stack  
2                comprises:  
3                    an entry number field;  
4                    a microinstruction pointer ( $\mu$ IP) field;  
5                    a back pointer field;  
6                    a retirement indicator field; and  
7                    a return pointer field.
- 1           3.    The processor of claim 2 in which the  $\mu$ IP field is  
2                14-bits wide.
- 1           4.    The processor of claim 3 in which the  $\mu$ IP field has  
2                a microinstruction pointer ( $\mu$ IP) pushed by a first  
3                microoperation ( $\mu$ Op) code and used by a second  $\mu$ Op code.
- 1           5.    The processor of claim 2 in which the back pointer  
2                field has a pointer to a next entry in the  $\mu$ IP stack for a  
3                micro-type of service ( $\mu$ TOS) bit to point to after a  $\mu$ Op.

1           6.    The processor of claim 2 in which the retirement  
2   indicator field has an indication of whether an entry has  
3   retired.

1           7.    The processor of claim 2 in the return pointer field  
2   a pointer to a location in a retirement stack to which an  
3   entry is copied after being retired.

1           8.    A method executed in a processor comprising:  
2                executing microcode ( $\mu$ code) stored in an out-of-  
3   order microinstruction pointer ( $\mu$ IP) stack; and  
4                manipulating the  $\mu$ IP stack with a set of  
5   microinstructions.

1           9.    The method of claim 8 in which the stack has an  
2   entry number field, a microinstruction pointer ( $\mu$ IP) field, a  
3   back pointer field, a retirement indicator field and a return  
4   pointer field.

1           10.   The method of claim 9 in which the  $\mu$ IP pointer field  
2   is 14-bits wide.

1           11.   The method of claim 10 in which the  $\mu$ IP pointer  
2   field has a microinstruction pointer ( $\mu$ IP) pushed by a first  
3   microoperation ( $\mu$ Op) code and used by a second  $\mu$ Op code.

1           12. The method of claim 9 in which the back pointer  
2 field has a pointer to a next entry in the  $\mu$ IP stack for a  
3 micro-type of service ( $\mu$ TOS) bit to point to after a  $\mu$ Op.

1           13. The method of claim 9 in which the retirement  
2 indicator field has an indication of whether an entry has  
3 retired.

1           14. The method of claim 9 in which the return pointer  
2 field contains a pointer to a location in a retirement stack  
3 to which an entry is copied after being retired.

1           15. The method of claim 9 in which manipulating  
2 comprises:  
3           pushing a next  $\mu$ IP on to the  $\mu$ IP stack; and  
4           using the next  $\mu$ IP in an intermediate field as a target  
5  $\mu$ IP in a jump operation.

1           16. The method of claim 9 in which manipulating  
2 comprises:  
3           taking a value of an intermediate field of a  
4 microoperation ( $\mu$ Op); and  
5           pushing the value on to the  $\mu$ IP stack.

1        17. The method of claim 9 in which manipulating  
2 comprises:

3        popping a value off the  $\mu$ IP stack; and  
4        replacing a current  $\mu$ Op intermediate field.

1        18. The method of claim 9 in which manipulating  
2 comprises:

3        popping a value off of the  $\mu$ IP stack; and  
4        jumping to that value.

1        19. The method of claim 9 in which manipulating  
2 comprises:

3        reading a value off the  $\mu$ IP stack; and  
4        replacing a  $\mu$ Op's intermediate field with the value.

1        20. The method of claim 9 in which manipulating  
2 comprises setting the  $\mu$ IP stack pointers to reset.

1        21. The method of claim 9 further comprising providing a  
2 set of pointers that point to different entries in the  $\mu$ IP  
3 stack.

1        22. The method of claim 21 in which the set of pointers  
2 includes a  $\mu$ TOS pointer that points to a top of the  $\mu$ IP stack.

1        23. The method of claim 21 in which the set of pointers  
2 includes a  $\mu$ Alloc pointer that points to a next allocated  
3 entry in the  $\mu$ IP stack.

1        24. The method of claim 21 in which the set of pointers  
2 includes a NextRet pointer that points to a next entry in the  
3  $\mu$ IP stack to be deallocated.

1        25. The method of claim 21 in which the set of pointers  
2 includes  $\mu$ RetTos pointer that points at a retired top of the  
3  $\mu$ IP stack.

1        26. The method of claim 8 in which the  $\mu$ OPs include an  
2 ms\_call  $\mu$ OP that takes a next  $\mu$ IP, pushes the next  $\mu$ IP on the  
3  $\mu$ IP stack, and uses the next  $\mu$ IP in an intermediate field as a  
4 target  $\mu$ IP of a jump.

1        27. The method of claim 8 in which the  $\mu$ OPs include an  
2 ms\_push  $\mu$ OP that takes a value in an intermediate field and  
3 pushes the value on the  $\mu$ IP stack.

1        28. The method of claim 8 in which the  $\mu$ OPs include an  
2 ms\_pop  $\mu$ OP that pops a value off the  $\mu$ IP stack and replaces  
3 the value with the  $\mu$ OP's intermediate field.

1        29. The method of claim 8 in which the  $\mu$ OPs include an  
2        `ms_return`  $\mu$ OP that pops a value off of the  $\mu$ IP stack and jumps  
3        to that  $\mu$ IP.

1        30. The method of claim 8 in which the  $\mu$ OPs include an  
2        `ms_tos_read`  $\mu$ OP that reads a value off the  $\mu$ IP stack and  
3        replaces this  $\mu$ OP's intermediate field.

1        31. The method of claim 8 in which the  $\mu$ OPs include an  
2        `ms_μip_stack_clear`  $\mu$ OP that sets the  $\mu$ IP stack pointers to  
3        reset.

1        32. A computer program product residing on a computer  
2        readable medium having instructions stored thereon which, when  
3        executed by the processor, cause the processor to:

4        execute microcode ( $\mu$ code) stored in an out-of-order  
5        microinstruction pointer ( $\mu$ IP) stack; and  
6        manipulate the  $\mu$ IP stack with a set of microinstructions.

1        33. The computer program product of claim 32 wherein  
2        instructions to manipulate further comprise instructions to:

3        push a next  $\mu$ IP on to the  $\mu$ IP stack; and  
4        use the next  $\mu$ IP in an intermediate field as a target  $\mu$ IP  
5        in a jump operation.

1        34. The computer program product of claim 32 wherein  
2 instructions to manipulate further comprise instructions to:  
3        take a value of an intermediate field of a microoperation  
4        ( $\mu$ Op); and  
      push the value on to the  $\mu$ IP stack.

1        35. The computer program product of claim 32 wherein  
2 instructions to manipulate further comprise instructions to:  
3        pop a value off the  $\mu$ IP stack; and  
4        replace a current  $\mu$ Op intermediate field with the value.

1        36. The computer program product of claim 32 wherein  
2 instructions to manipulate further comprise instructions to:  
3        pop a value off of the  $\mu$ IP stack; and  
4        jump to that value.

1        37. The computer program product of claim 32 wherein  
2 instructions to manipulate further comprise instructions to:  
3        read a value off the  $\mu$ IP stack; and  
4        replace a  $\mu$ Op's intermediate field with the value.

1        38. The computer program product of claim 32 wherein  
2 instructions to manipulate further comprise instructions to:  
3        set the  $\mu$ IP stack pointers to reset.